San José State University
Department of Applied Data Science

DATA 200
Computational Programming for Analytics

Spring 2024
Instructor: Ron Mak

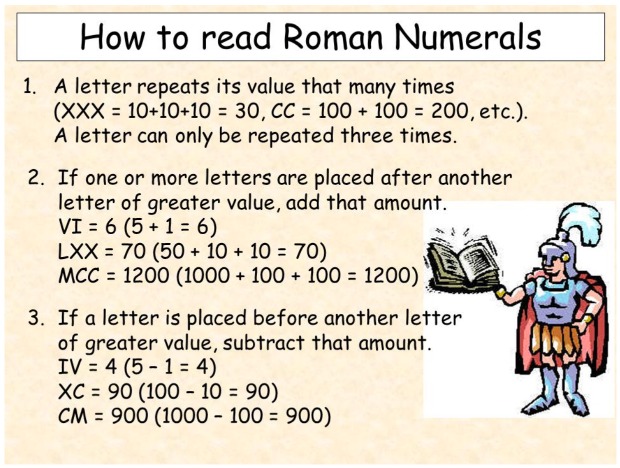
Assignment #9

Assigned: Thursday, March 28
Due: Thursday, April 11 at 5:30 PM
Points: Maximum 250

You may choose a partner to work together on this assignment. Turn in only one assignment and clearly state who the partners are. Both of you will receive the same score.

Roman numeral class
You will practice creating a Python class that has public and private attributes and methods, properties, and overloaded special methods and operators.

For a refresher on Roman numerals, see https://en.wikipedia.org/wiki/Roman_numerals
Read up to but not including the section "Other forms".



How to read Roman Numerals

1. A letter repeats its value that many times (XXX = 10+10+10 = 30, CC = 100 + 100 = 200, etc.). A letter can only be repeated three times.
2. If one or more letters are placed after another letter of greater value, add that amount.
   VI = 6 (5 + 1 = 6)
   LXX = 70 (50 + 10 + 10 = 70)
   MCC = 1200 (1000 + 100 + 100 = 1200)
3. If a letter is placed before another letter of greater value, subtract that amount.
   IV = 4 (5 - 1 = 4)
   XC = 90 (100 - 10 = 90)
   CM = 900 (1000 - 100 = 900)

Class `RomanNumeral`
Design and implement a Python class **`RomanNumeral`** in module **`roman`** (source file **`roman.py`**) that performs arithmetic operations on Roman numerals. This class must have:

- Private attributes **`_roman`** (a string) and **`_value`** (an integer) that store the Roman numeral string (such as **`'MCMLXVIII'`**) and the corresponding decimal value (such as 1968) of each **`RomanNumeral`** object.
- Private methods **`_to_roman()`** and **`_to_value()`** that convert between the string and integer values of a **`RomanNumeral`** object and thereby set the values of member variables **`_roman`** and **`_value`**. These two attributes should always be synchronized to represent the same value.

- Constructor method `__init__()` one that takes a single parameter that can be either a Roman numeral string such as `'MMXXIII'` or an integer such as `2023`. If the parameter value is a string, store it in `_roman`, convert it to its integer value, and store the value in `_value`. If the parameter value is an integer, store it in `_value`, convert it to its Roman numeral string, and store the string in `_roman`.
- Public read-only properties that return a `RomanNumeral` object's string and integer values.
- Overloaded arithmetic operators `+ - *` and `//` that enable direct arithmetic operations with Roman numerals. Each operator must return a `RomanNumeral` object and the operation must not change the value of either operand. Roman numerals do floor division.
- Overloaded special method `__str__()` that returns a string for a `RomanNumeral` object for printing in the form **[***string***:***value***]** . For example:

```
[MCMLXVIII:1968]
```

- Overloaded special method `__repr__()` that returns a string that represents a `RomanNumeral` object in the form **RomanNumeral(roman='***string***', value=***value***)**. For example:

```
RomanNumeral(roman='MCMLXVIII', value=1968)
```

You may assume for this assignment that the values of the Roman numerals range from 1 through 3999. (Did the ancient Romans have a zero or negative numbers?)

Tip: Do the operations using integer arithmetic on the operands' values and then convert the results to Roman numeral strings.

Test the class

Jupyter notebook `RomanNumeralTests.ipynb` contains four problems to test your `RomanNumeral` class.

http://www.cs.sjsu.edu/~mak/DATA200/assignments/9/RomanNumeralTests.ipynb

The tests include reading the input text file `expressions.txt` and performing the operations:

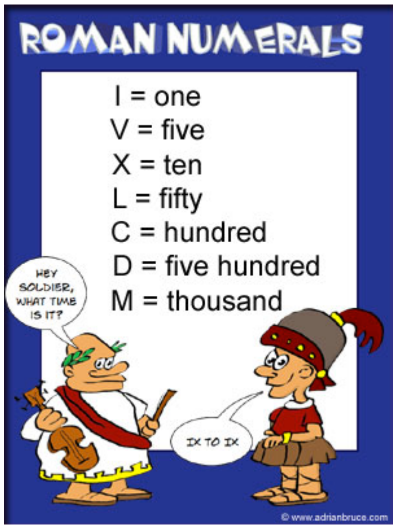
```
MCMLXIII + LVI
MMI - XXXIII
LIII * XXXIII
MMI // XXXIII
```

http://www.cs.sjsu.edu/~mak/DATA200/assignments/9/expressions.txt

Tip: Use regular expressions to extract and capture the two Roman numerals and the operator.

The file contains simple two-operand arithmetic expressions with Roman numerals. The operator is either `+` `-` `*` or `//`. The function should read each expression, perform the operation, and print the expression and its result similar to:

```
[MCMLXIII:1963] + [LVI:56] = [MMXIX:2019]
[MMI:2001] - [XXXIII:33] = [MCMLXVIII:1968]
[LIII:53] * [XXXIII:33] = [MDCCXLIX:1749]
[MMI:2001] // [XXXIII:33] = [LX:60]
```



You may assume that all the Roman numerals in the input are in upper case, and that there are no input errors. Therefore, for this assignment, you do not need to do input error checking.

What to submit
The completed test Jupyter notebook.

Rubric

Criteria	Max points
Class components	**110**
• Constructor `__init__()`	• 10
• Private method `_to_roman`	• 10
• Private method `_to_value`	• 10
• Read-only property for Roman string	• 10
• Read-only property for integer value	• 10
• Overloaded `+` operator	• 10
• Overloaded `-` operator	• 10
• Overloaded `*` operator	• 10
• Overloaded `//` operator	• 10
• Overloaded `__repl__()` special method	• 10
• Overloaded `__str__()` special method	• 10
Good program design	**40**
• Use of function docstrings.	• 10
• Good names and internal comments.	• 10
• Well-designed class.	• 20
Problems	**100**
• Problem 1	• 20
• Problem 2	• 20
• Problem 3	• 20
• Problem 4	• 40